**CSCI 1301 Lecture 6**

Key points of this lecture:

1. Put it all together
2. Write a simple java application

  All java programs are contained in classes. The naming convention of any class starts with an uppercase letter. If the class name includes more than one word, the naming conventions calls for using uppercase for each of the starting letters of all the words forming the class name. Example for one word class name: Welcome, Student, Employee, Project, etc. Examples of multiword class names: HelloWorld, MyFirstProject, BankAccount, etc. That supports readability.   
  
For the scope of this course, most of our class definitions will start with two reserved words "public" and "class" followed by the actual class name that you selected. The file will be saved as the name of the public class you just created with the file extension ".java". For example, if your class name is HelloWorld then it will be saved as HelloWorld.java. Therefore, java program files can have only one public class. Following the class name, we use opening and closing curly braces to indicate the beginning and the ending of our class. Everything that makes up the class must be contained between those opening and closing curly braces. That is referred to as the body of the class. For example:   
  
**public class MyFirstJavaProgram  
{  
 ...  
 ...  
}**   
  
Java programs' starting point is the main method. For the scope of our course, most of our java programs will be one file of a java program with the main method declared and defined inside the body of our class. The main method is defined with the following java code:   
  
**public static void main(String args[])  
{  
 ...  
 ...  
}**   
  
Also similar to the class body, anything inside the opening and closing curly braces for the main method is the body of the main method. Now, to put these two together we create a java program called MyFirstJavaProgram.java and it will contain the following code:   
  
**public class MyFirstJavaProgram  
{  
 public static void main(String args[])  
 {  
  ...  
  ...  
 }  
}**   
  
The indentations in the code above are for readability only. Java is not like Python where indentation is part of the code and it has a meaning. In java, white spaces are ignored and alignments or indentations are used for readability. To include a simple java statement to print some words on the screen we write the following java code: System.out.println("! This is my first java program."); So, if I want to create a java program that prints the words "Hello! This is my first java program." We put all the pieces described above together as the following:   
  
**public class MyFirstJavaProgram  
{  
 public static void main(String args[])  
 {  
  System.out.println("Hello! This is my first java program.");  
 }  
}**   
  
We need to save this code in a java file as "MyFirstJavaProgram.java". The compiling process is used to check the code you wrote against the java rules. To compile a java program, from the command line you execute the following command: javac MyFirstJavaProgram.java If there are errors in your code the compiler will point them out and give you a feedback to help you identify and resolve the errors. Those errors are referred to as "syntax errors". There are three different types of errors:

1. Syntax error
2. runtime error
3. logical errors

Syntax errors are captured by the compiler during the compiling process. Runtime errors are when the program crashes soon after it starts runny. Logical errors are when the final outcome or results of the java program are not accurate or are not correct (example when including spelling mistakes or wrong results of the performed calculations).   
  
Compile, resolve errors and program problems and run your program. To run the program you execute the following command line:" java MyFirstJavaProgram Note that we didn't include the file extension when trying to run the program. Use the file extension only with the "javac" command to compile the java program. Use comments to document your program.   
  
**Appropriate variable declaration**  
The variable declaration is the specification of the data type and the variable name. Example of variable declaration:   
int age;  
int stuNumber;  
char grade;  
double gpa;  
  
Multiple variables, of the same type, can be declared in one statement. For example: int age, stuNumber, counter;  
  
The variable initialization is the step where we give a value, or initialize, the variable. For example:  
age = 21;  
stuNumber = 26;  
grade = 'A';  
gpa = 3.5;  
  
If we know the initial value of the variable we can declare and initialize the variable in one statement. For example:  
int stuNumber = 26;  
  
  
The statement above is similar to the following two statements:  
int stuNumber;  
stuNumber = 26;  
  
  
Also multiple variables of the same type can be declared and initialized in one statement. For example:  
int age = 21, stuNumber = 26, maxGrade = 100;  
  
Please also review the following topics:

* Arithmetic Operations
* Operator Precedence
* Increment and decrement

**Interactive programming:**

Scanner class is an easy way to receive various types of inputs to the program. System.in represents input from the keyboard. The following is a statement that creates an object that can access class Scanner and read entries from the Keyboard:   
Scanner userInput = new Scanner (System.in);   
  
The new object created, userInput, can access the Scanner class and use, invoke, its methods. Input methods from Scanner class such as nextLine(), nextInt(), nextDouble(). As indicated by the type, nextInt(); will assign the type int to the entry, nextDouble(); will receive a double type, nextChar(); will receive a character type, and nextLine(); will receive a String.   
  
The Scanner class is part of java.Util class library and must be imported to the program for the program to be able to access and use Scanner class. Consider the following sample program to illustrate interactive programming and the use of Scanner class:   
  
**import java.util.Scanner;  
public class MyTime  
{  
 public static void main (String[] args)  
 {  
  int distance;  
  double speed, time;  
  
  Scanner enteredData = new Scanner (System.in);  
  System.out.print ("Enter the distance in miles: ");  
  distance = enteredData.nextInt();  
  System.out.print ("Enter the speed: ");  
  speed = enteredData.nextDouble();  
  time = distance / speed;  
  System.out.println ("The time required to reach destination is: " + time);  
 }  
}**